

✓ Hissa al otaibi

✓

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import warnings
from sklearn.preprocessing import LabelEncoder
import re
import nltk
from nltk.corpus import stopwords
from sklearn.ensemble import RandomForestClassifier
```

```
df = pd.read_csv('/kaggle/input/datasets/areejhaddaj/sentiment/clean_dataset (1).csv')
df = df.dropna(subset=['clean_text'])
df.head(7)
```

	sentiment	clean_text
0	negative	sooo sad miss san diego
1	negative	boss bulli
2	negative	interview leav alon
3	negative	son couldnt put releas already bought
4	positive	feed babi fun smile coo
5	positive	journey wow u becam cooler hehe possibl
6	positive	realli realli like song love stori taylor swift

```
df.isnull().sum()
```

```
sentiment    0
clean_text   0
dtype: int64
```

```
nltk.download('stopwords', quiet=True)
stop_words = set(stopwords.words('english'))

def advanced_preprocessing(text):
    if not isinstance(text, str):
        return ""

    text = text.lower()

    # Normalization
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    text = re.sub(r'\@w+|\#','', text)
    text = re.sub(r'^a-z\s','', text)

    words = text.split()
    words = [w for w in words if w not in stop_words]

    return " ".join(words)

df['clean_text'] = df['clean_text'].apply(advanced_preprocessing)
```

```
X = df['clean_text']
y = df['sentiment']
```

```
le = LabelEncoder()
df['sentiment'] = le.fit_transform(df['sentiment'])
```

```
X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

vectorizer = TfidfVectorizer(max_features=5000)
X_train = vectorizer.fit_transform(X_train_raw)
X_test = vectorizer.transform(X_test_raw)
```

```
name = "Random Forest"
rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)

print(f"\n{' '*40}\n Model: {name}\n{' '*40}")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2%}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
```

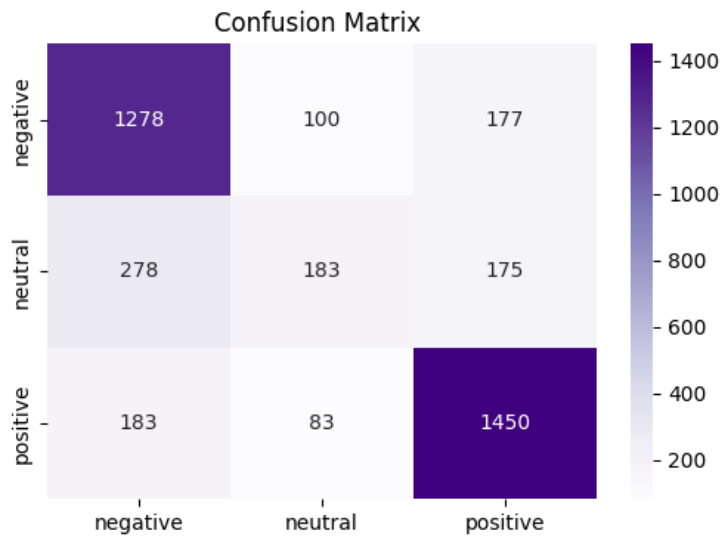
```
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Purples', xticklabels=le.classes_, yticklabels=le.classes_)
plt.title(f'Confusion Matrix ')
plt.show()
```

```
=====
Model: Random Forest
=====
```

Accuracy: 74.51%

Classification Report:

	precision	recall	f1-score	support
negative	0.73	0.82	0.78	1555
neutral	0.50	0.29	0.37	636
positive	0.80	0.84	0.82	1716
accuracy			0.75	3907
macro avg	0.68	0.65	0.66	3907
weighted avg	0.73	0.75	0.73	3907



```
name = "Naive Bayes"
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
y_pred = nb_model.predict(X_test)

print(f"\n{' '*40}\n Model: {name}\n{' '*40}")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2%}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

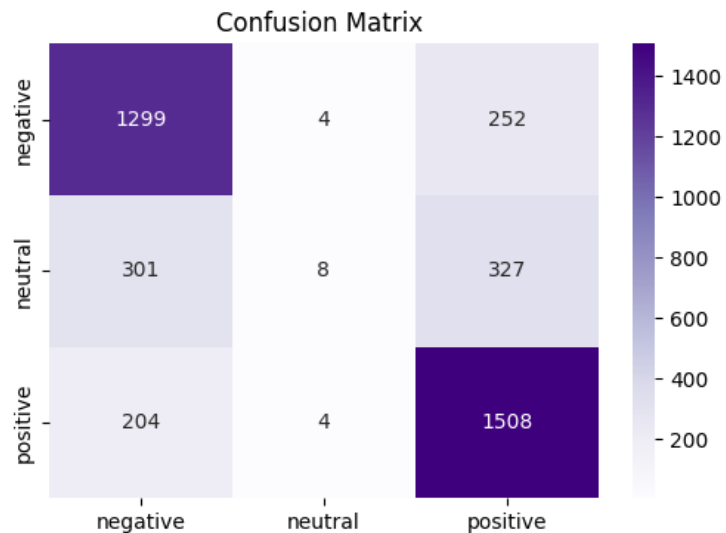
```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Purples', xticklabels=le.classes_, yticklabels=le.classes_)
plt.title(f'Confusion Matrix ')
plt.show()
```

```
=====
Model: Naive Bayes
=====
Accuracy: 72.05%
```

```
Classification Report:
              precision    recall  f1-score   support

 negative         0.72         0.84         0.77        1555
   neutral         0.50         0.01         0.02         636
   positive         0.72         0.88         0.79        1716

 accuracy          0.72         0.72         0.72        3907
   macro avg         0.65         0.58         0.53        3907
   weighted avg         0.69         0.72         0.66        3907
```



```
name = "Support Vector Machine"
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)

print(f"\n{' '*40}\n Model: {name}\n{' '*40}")
```

```
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2%}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Purples', xticklabels=le.classes_, yticklabels=le.classes_)
plt.title(f'Confusion Matrix ')
plt.show()
```

```
=====
Model: Support Vector Machine
=====
Accuracy: 73.84%
```

```
Classification Report:
              precision    recall  f1-score   support

negative     0.73         0.83         0.78        1555
neutral       0.42         0.22         0.29         636
positive     0.80         0.84         0.82        1716

accuracy          0.74         0.74         0.74        3907
macro avg         0.65         0.63         0.63        3907
weighted avg      0.71         0.74         0.72        3907
```

